On Local Domain Symmetry for Model Expansion

Jo Devriendt[†], Bart Bogaerts[‡,†], Maurice Bruynooghe[†], Marc Denecker[†]

[†] *KU Leuven – University of Leuven, Celestijnenlaan 200A, Leuven, Belgium*
(*e-mail:* firstname.lastname@cs.kuleuven.be)
[‡] *Helsinki Institute for Information Technology HIIT, Aalto University, FI-00076 AALTO, Finland*

submitted 1 January 2003; revised 1 January 2003; accepted 1 January 2003

Abstract

Symmetry in combinatorial problems is an extensively studied topic. We continue this research in the context of model expansion problems, with the aim of automating the workflow of detecting and breaking symmetry. We focus on *local domain symmetry*, which is induced by permutations of domain elements, and which can be detected on a first-order level. As such, our work is a continuation of the symmetry exploitation techniques of model generation systems, while it differs from more recent symmetry breaking techniques in answer set programming which detect symmetry on ground programs. Our main contributions are sufficient conditions for symmetry of model expansion problems, the identification of *local domain interchangeability*, which can often be broken completely, and efficient symmetry detection algorithms for both local domain interchangeability as well as local domain symmetry in general. Our approach is implemented in the model expansion system IDP, and we present experimental results showcasing the strong and weak points of our approach compared to SBASS, a symmetry breaking technique for answer set programming.

Under consideration for acceptance in TPLP.

1 Introduction

Many problems exhibit symmetry. For instance, the set of trucks in a routing problem is interchangeable, a chess board can be mirrored onto itself, an input graph has non-trivial automorphisms, etc. It is a well-known burden of combinatorial search engines that they visit each of the (potentially exponentially many) symmetric areas of their search space, and hence waste valuable time rediscovering already known information. In order to solve this problem, research on symmetry has been extensive, especially in the constraint programming (CP) (Gent et al. 2006) and satisfiability solving (SAT) community (Sakallah 2009).

For logic-based systems, much work has been done in the context of theorem proving and finite model generation systems (Zhang and Zhang 1995; Audemard and Benhamou 2002; Claessen and Sörensson 2003; Torlak and Jackson 2007). With the advent of answer set programming (ASP), interest in symmetry for logics is renewed (Drescher et al. 2011).

In this paper, we continue research on symmetry in classical logic, with a focus on symmetry for model expansion problems. We propose the notion of *local domain symmetry* in Section 3, a common form of symmetry stemming from permutations of domain elements. We show in Section 4 how *local domain interchangeability*, a particular type of local domain symmetry, can be broken completely with a linear number of symmetry breaking constraints. Section 5 gives a detection algorithm for local domain symmetry in general, and for local domain interchangeability in particular. These detection algorithms operate on a first-order level, hence they avoid the computational blow-up of a *ground* theory. In Section 6, we experimentally compare our approach to

state-of-the-art ASP symmetry breaking which performs symmetry detection on the ground theory. These experiments confirm that symmetry detection at the first order level is indeed faster, and in some cases, we achieve stronger symmetry breaking. However, there are also cases where less symmetry is detected. We conclude in Section 7. Proofs are postponed to Appendix A.

2 Preliminaries

We assume familiarity with the basics of first-order logic. A *vocabulary* Σ is a set of *predicate symbols* P/n of arity $n \geq 0$ and *function symbols* f/n of arity $n \geq 0$. Often, we will simply refer to a *symbol* $S/n \in \Sigma$, which represents an n-ary predicate or function symbol. *Variables*, *terms*, *atoms*, *quantifiers*, *formulas* and *theories* are defined as usual (Enderton 2001). A Σ-theory T is a theory with vocabulary Σ, i.e., such that the free symbols of T are all in Σ.

Slightly deviating from the standard presentation of first-order logic, we consider both variables and constants to be function symbols of arity 0, as they both serve to identify a single domain element. Without loss of generality, we assume variables are *renamed apart*, i.e., each variable is bound by at most one quantifier. This simplifies the presentation of our results.

A Σ-*structure* I consists of a domain D and an *interpretation* to each symbol in Σ. For each n-ary predicate symbol P, interpretation P^I is an n-ary relation on D, i.e., $P^I \subseteq D^n$. For an n-ary function f, interpretation f^I is an n-ary function $D^n \to D$. If Σ-structure I and Σ'-structure I' have the same domain D and $\Sigma \cap \Sigma' = \emptyset$, $I \sqcup I'$ is the $\Sigma \cup \Sigma'$-structure over D that interprets all symbols in Σ according to I and all symbols in Σ' according to I'. If I is a Σ-structure with domain D, x a 0-ary function symbol, and $d \in D$, then we use $I[x : d]$ for the $\Sigma \cup \{x\}$-structure that equals I, except for interpreting x by d. Because variables are considered 0-ary function symbols as well, the notion of "variable assignment", which is used in the classical presentation of first-order logic, is subsumed by our notion of structure extension $I[x : d]$.

The *value* t^I of a term t in a structure I is a domain element $d \in D$, and is defined as usual (assuming I interprets all symbols in t). The *truth value* φ^I of a formula φ in a structure I is either true or false, and is also defined as usual. A Σ-structure I is a *model* of a Σ-theory T (written as $I \models T$) if for each formula φ in T, φ^I is true. Many combinatorial problems can be conveniently modelled as a *model expansion* problem $MX(T, I_{in})$, where T is a Σ-theory T and I_{in} is a Σ_{in}-structure with $\Sigma_{in} \subseteq \Sigma$. We refer to Σ_{in} as the *input vocabulary*, and $\Sigma_{out} = \Sigma \setminus \Sigma_{in}$ as the *output vocabulary*. A *solution* to a model expansion problem $MX(T, I_{in})$ with output vocabulary Σ_{out} is a Σ_{out}-structure I_{out} (sharing I_{in}'s domain) such that $I_{in} \sqcup I_{out} \models T$; $I_{in} \sqcup I_{out}$ is a model to T that *expands* I_{in}.

3 Symmetries

Throughout this section, we assume a fixed domain D and use Γ_D to refer to the set of all structures with domain D.

As a running example, we use a simple graph coloring problem.

Example 3.1
Let Σ_{gc} be the vocabulary consisting of predicate symbols $V/1$, $C/1$, $Edge/2$ and a function symbol $Color/1$. A valid colored graph is expressed by the theory T_{gc}:

$$\forall x_1\ y_1 : Edge(x_1, y_1) \Rightarrow (Color(x_1) \neq Color(y_1))$$
$$\forall x_2\ y_2 : Edge(x_2, y_2) \Rightarrow V(x_2) \wedge V(y_2)$$
$$\forall x_3 : C(Color(x_3))$$

Let $\Sigma_{gcin} = \Sigma_{gc} \setminus \{Color/1\}$. Input data containing vertices, colors and a graph is expressed as a Σ_{gcin}-structure I_{gcin} with domain $D = \{t,u,v,w,r,g,b\}$, with interpretations

$$V^{I_{gcin}} = \{t,u,v,w\} \;\; Edge^{I_{gcin}} = \{(t,u),(u,v),(v,w),(w,t)\} \;\; C^{I_{gcin}} = \{r,g,b\}.$$

The model expansion problem $MX(T_{gc}, I_{gcin})$ now consists of finding a $\Sigma_{gcout} = \{Color/1\}$-structure I_{gcout} such that $I_{gcin} \sqcup I_{gcout} \models T_{gc}$. We let I_{gcout} contain the interpretation

$$Color^{I_{gcout}} = t \mapsto r, u \mapsto g, v \mapsto b, w \mapsto g, r \mapsto r, g \mapsto g, b \mapsto b$$

which represents a valid coloring of the input graph. Indeed, $I_{gc} = I_{gcin} \sqcup I_{gcout} \models T_{gc}$. ▽

3.1 Symmetry of a theory

Definition 3.2 (*Symmetry*)
A mapping $\sigma\colon \Gamma_D \to \Gamma_D$ is a *structure transformation.* A structure transformation σ is a *symmetry* for Σ-theory T if for all Σ-structures $I \in \Gamma_D$, $I \models T$ iff $\sigma(I) \models T$.

The set of symmetries of T forms a group under composition ($\circ$). In this paper, we study how to detect and exploit symmetries. Detecting all symmetries of a theory is computationally at least as hard as deciding whether the theory is satisfiable (if not, all structure transformations are symmetries). Instead, we focus on symmetries that can be detected by means of syntactical analysis, and that are induced by permutations of domain elements.

Definition 3.3 (*Domain permutation*)
A bijection $\pi\colon D \to D$ is a *domain permutation*. A domain permutation *induces a structure transformation* σ_π: for each predicate symbol P/n, $(\pi(d_1),\ldots,\pi(d_n)) \in P^{\sigma_\pi(I)}$ iff $(d_1,\ldots,d_n) \in P^I$, and for each function symbol f/n, $f^{\sigma_\pi(I)}(\pi(d_1),\ldots,\pi(d_n)) = \pi(d)$ iff $f^I(d_1,\ldots,d_n) = d$.

Proposition 3.4
Any structure transformation induced by a domain permutation is a symmetry for any theory.

We call this type of symmetry induced by only a domain permutation *global domain symmetry*.

We use cycle notation to compactly represent permutations, e.g., $(a\ b\ c)(d\ e)$ is a permutation that maps element a to b, b to c, c to a, swaps d and e, and maps any other element to itself.

Example 3.5 (*Example 3.1 continued*)
The domain permutation $(v\ r)$ induces a global domain symmetry $\sigma_{(v\ r)}$ of T_{gc}. $\sigma_{(v\ r)}(I_{gc})$ gives

$$D = \{t,u,v,w,r,g,b\} \quad V^{\sigma_{(v\ r)}(I_{gc})} = \{t,u,r,w\} \quad Edge^{\sigma_{(v\ r)}(I_{gc})} = \{(t,u),(u,r),(r,w),(w,t)\}$$
$$C^{\sigma_{(v\ r)}(I_{gc})} = \{v,g,b\} \quad Color^{\sigma_{(v\ r)}(I_{gc})} = t \mapsto v, u \mapsto g, r \mapsto b, w \mapsto g, v \mapsto v, g \mapsto g, b \mapsto b,$$

which is still a model of T_{gc} (though r now acts as a vertex and v as a color). ▽

Finite model generators such as KODKOD (Torlak and Jackson 2007) or SEM (Zhang and Zhang 1995) focus on the task of generating a model with a given domain for a given theory. Since every domain permutation induces a global domain symmetry, these systems have mechanisms to cope with global domain symmetry.

However, a global domain symmetry σ_π is a rather restrictive concept as it applies π on every argument of every tuple in every interpretation of a structure. A larger class of transformations can be described when π is only applied locally. For example, one could apply π only on the

interpretation of some symbols, or even more fine-grained, only on some of the arguments in the tuples of an interpretation. Given a predicate or function symbol S/n, we use $S|i$ with $1 \leq i \leq n$ to denote the i^{th} *argument position* of S; if S is a function symbol, we use $S|0$ for the output argument of S. Note that variables, being treated as function symbols, also form argument positions.

Definition 3.6 (*Structure transformation induced by* A, π)
Let π be a domain permutation and A a set of argument positions. The structure transformation σ^A_π *induced by* A, π is defined by

$$(\tau_{P|1}(d_1), \ldots, \tau_{P|n}(d_n)) \in P^{\sigma^A_\pi(I)} \text{ iff } (d_1, \ldots, d_n) \in P^I$$
$$f^{\sigma^A_\pi(I)}(\tau_{f|1}(d_1), \ldots, \tau_{f|n}(d_n)) = \tau_{f|0}(d_0) \text{ iff } f^I(d_1, \ldots, d_n) = d_0$$

where $\tau_{S|i}(d) = \pi(d)$ if $S|i \in A$ and $\tau_{S|i}(d) = d$ otherwise.

Thus, for each domain tuple in the interpretation of a symbol S, the structure transformation induced by A, π only applies π to domain elements that occur at an index i corresponding to an argument position $S|i \in A$. Note that if A contains argument positions over symbols S not interpreted by I (e.g., variable symbols), those argument positions are simply ignored by σ^A_π.

Definition 3.7 (*Local domain symmetry*)
Let T be a theory. A *local domain symmetry* for T is a structure transformation induced by a set of argument positions A and a domain permutation π, that also is a symmetry for T.

A global domain symmetry σ_π for a Σ-theory is a local domain symmetry σ^A_π where A includes all argument positions of all symbols in Σ. As such, local domain symmetry is a generalization of global domain symmetry, and allows us to detect and exploit more symmetry. However, not all A, π-induced structure transformations are symmetries. Below, we propose a syntactic criterion to identify a set of argument positions A that guarantees that σ^A_π is a symmetry for a given theory. Intuitively, the criterion can be formulated as follows: whenever a term $f(\ldots)$ occurs as the i'th argument in a predicate or function symbol S, then $f|0 \in A$ if and only if $S|i \in A$.

Definition 3.8
Let T be a theory. Assume $f|0$ and $S|i$ are argument positions with S either a predicate or a function symbol. We call $f|0$ and $S|i$ *directly connected by* T if one of the following holds:

- an expression $S(t_1, \ldots, t_{i-1}, f(\bar{t}'), t_{i+1}, \ldots, t_n)$ occurs in T, or
- $i = 0$ and an expression $S(\bar{t}) = f(\bar{t}')$ occurs in T.

A set A of argument positions is *connectively closed under* T if for each $S|i \in A$, each argument position $R|j$ directly connected to $S|i$ by T, is also in A.

Example 3.9 (*Example 3.1 continued*)
According to the first formula in T_{gc}, $x_1|0$ is directly connected to $Edge|1$ and $Color|1$, while $y_1|0$ is directly connected to $Edge|2$ and $Color|1$. Analyzing all formulas, we find the following two sets are connectively closed under T_{gc}: $A = \{C|1, Color|0\}$ and $B = \{V|1, Edge|1, Edge|2, Color|1, x_1|0, y_1|0, x_2|0, y_2|0, x_3|0\}$.

Applying the induced structure transformation $\sigma^A_{(v\ r)}$ on I_{gc} gives

$$D = \{t,u,v,w,r,g,b\} \quad V^{\sigma^A_{(v\ r)}(I_{gc})} = \{t,u,v,w\} \quad Edge^{\sigma^A_{(v\ r)}(I_{gc})} = \{(t,u),(u,v),(v,w),(w,t)\}$$
$$C^{\sigma^A_{(v\ r)}(I_{gc})} = \{v,g,b\} \quad Color^{\sigma^A_{(v\ r)}(I_{gc})} = t \mapsto v, u \mapsto g, v \mapsto b, w \mapsto g, r \mapsto v, g \mapsto g, b \mapsto b$$

which is also a model of T_{gc} (here, domain element v serves both as a vertex and a color). ▽

Theorem 3.10 (*Local domain symmetry condition*)
Let Σ be a vocabulary, T a theory over Σ, π a domain permutation and A a set of argument positions. If A is connectively closed under T, then σ_π^A is a local domain symmetry of T.

This theorem is useful when detecting symmetry for model expansion problems with an empty input structure, but it will also prove useful for non-empty input structures.

Example 3.11 (*Example 3.9 continued*)
The argument position set A induces local domain symmetries that correspond to permuting the colors of a graph coloring problem, while B induces symmetry on the vertices and $A \cup B$ induces global domain symmetries. ▽

3.2 Symmetry for model expansion

Recall that a model expansion problem $MX(T, I_{in})$ consists of finding structures I_{out} such that $I_{in} \sqcup I_{out} \models T$.

Definition 3.12 (*Symmetry for MX*)
Let $MX(T, I_{in})$ be a model expansion problem with output vocabulary Σ_{out}, and let $\Gamma_D^{\Sigma_{out}}$ be the set of Σ_{out}-structures with domain D. A structure transformation $\sigma\colon \Gamma_D^{\Sigma_{out}} \to \Gamma_D^{\Sigma_{out}}$ is a symmetry of $MX(T, I_{in})$ if for each $I_{out} \in \Gamma_D^{\Sigma_{out}}$, $I_{in} \sqcup I_{out} \models T$ iff $I_{in} \sqcup \sigma(I_{out}) \models T$.

Analogous to Definition 3.6, a domain permutation π and argument position set A induce a structure transformation σ_π^A on $\Gamma_D^{\Sigma_{out}}$. We call σ_π^A a *local domain symmetry of* $MX(T, I_{in})$ if σ_π^A is a symmetry of $MX(T, I_{in})$.

Example 3.13 (*Example 3.1 continued*)
Let A be the argument position set $\{V|1, Edge|1, Edge|2, Color|1, x_1|0, y_1|0, x_2|0, y_2|0, x_3|0\}$. Observe that A is connectively closed under T_{gc} and that the induced structure transformation $\sigma^A_{(t\ u\ v\ w)}$ is a local domain symmetry of $MX(T_{gc}, I_{gcin})$. However, connectively closedness under the theory is neither a sufficient nor a necessary condition for an A, π-induced structure transformation to be a symmetry of a model expansion problem.

For instance, argument position set $B = \{Edge|1, Edge|2, Color|1, x_1|0, y_1|0, x_3|0\}$ is not connectively closed under T_{gc}, though $\sigma^B_{(t\ u\ v\ w)}$ is still a symmetry of $MX(T_{gc}, I_{gcin})$.

Moreover, since A is connectively closed, $\sigma^A_{(v\ r)}$ is a local domain symmetry of T_{gc}, but it is not a symmetry of $MX(T_{gc}, I_{gcin})$. Indeed,

$$Color^{\sigma^A_{(v\ r)}(I_{gcout})} = t \mapsto v, u \mapsto g, v \mapsto b, w \mapsto g, r \mapsto v, g \mapsto g, b \mapsto b$$

maps t and r to node v, which is not consistent with $\forall x_3 : C(Color(x_3))$ and $C^{I_{gcin}}$. ▽

The above example shows that for model expansion, local domain symmetries are useful, but Theorem 3.10 does not suffice to identify them. Below, we give a sufficient condition for A, π-induced structure transformations to be local domain symmetries of a model expansion problem. For this, we require the notion of a *decomposition*.

Definition 3.14
Let $MX(T, I_{in})$ be a model expansion problem with input vocabulary Σ_{in}. Also, let T^* be equal to T with each occurrence of a symbol $S \in \Sigma_{in}$ replaced by a unique new copy S_i, let Σ^*_{in} be the vocabulary containing all copy symbols S_i, and let I^*_{in} be the Σ^*_{in}-structure where for each copy S_i, $S_i^{I^*_{in}} = S^{I_{in}}$. We call $MX(T^*, I^*_{in})$ the *decomposition* of $MX(T, I_{in})$.

It is clear that a model expansion problem and its decomposition have the same solutions, as they have the same output vocabulary and as each occurrence of a copy S_i in T^* imposes the same constraints on models for T^* as S did for T (since $S_i^{I_{in}^*} = S^{I_{in}}$).

Example 3.15 (*Example 3.13 continued*)
Let $MX(T_{gc}^*, I_{gcin}^*)$ be the decomposition of $MX(T_{gc}, I_{gcin})$. T_{gc}^* consists of

$$\forall x_1\ y_1 : Edge_1(x_1, y_1) \Rightarrow (Color(x_1) \neq Color(y_1))$$
$$\forall x_2\ y_2 : Edge_2(x_2, y_2) \Rightarrow V_1(x_2) \wedge V_2(y_2)$$
$$\forall x_3 : C_1(Color(x_3))$$

▽

Theorem 3.16 (*Local domain symmetry condition for MX*)
Let $MX(T, I_{in})$ be a model expansion problem with decomposition $MX(T^*, I_{in}^*)$. If A is connectively closed under T^* and $\sigma_\pi^A(I_{in}^*) = I_{in}^*$ then σ_π^A is a symmetry for $MX(T, I_{in})$.

Example 3.17 (*Example 3.15 continued*)
Argument position set $A = \{Edge_1|1, Edge_1|2, Color|1, x_1|0, y_1|0, x_3|0\}$ is connectively closed under T_{gc}^*, and $\sigma_{(t\ u\ v\ w)}^A(I_{gcin}^*) = I_{gcin}^*$. Thus, $\sigma_{(t\ u\ v\ w)}^A$ is a symmetry of $MX(T_{gc}, I_{gcin})$, exploiting cyclicity of the input graph. However, $\sigma_{(t\ u)}^A$ is not a symmetry of $MX(T_{gc}, I_{gcin})$, as the input interpretation of $Edge_1$ is not preserved by swapping t and u. ▽

Note that if $\Sigma_{in} = \emptyset$, the conditions of Theorem 3.16 degenerate into the conditions of Theorem 3.10. Also, for σ_π^A satisfying Theorem 3.16, A typically contains argument positions over both Σ_{out} and the decomposed Σ_{in}^* (as well as over variables). Lastly, the requirement that A is connectively closed under T^* is weaker than being closed under T. For example, let T be $P(f) \vee P(g)$, with only P interpreted by the input structure. The only connectively closed set under T is $\{P|1, f|0, g|0\}$. However, under the corresponding decomposition theory $T^* = P_1(f) \vee P_2(g)$, there are three connectively closed sets: $\{P_1|1, f|0\}$, $\{P_2|1, g|0\}$ and their union.

3.3 Subdomain interchangeability

Local domain symmetries for a theory T can be identified by computing argument position sets A that are connectively closed. Then, as mentioned in Section 3.1, any permutation π of the domain D gives rise to a local domain symmetry σ_π^A. For model expansion, σ_π^A must preserve the input structure, so not all π are guaranteed to induce symmetry. However, given a suitable set of argument positions A, a subdomain $\delta \subseteq D$ might exist for which any permutation of δ induces a symmetry of the model expansion problem.

Definition 3.18 (*A-interchangeable subdomain*)
Let $MX(T, I_{in})$ be a model expansion problem, A a set of argument positions and δ a subset of the domain. δ is an *A-interchangeable subdomain* if for every permutation π over δ, the structure transformation σ_π^A induced by A, π is a local domain symmetry for $MX(T, I_{in})$. The *subdomain interchangeability group* $\mathbb{G}_\delta^A$ is the group of all local domain symmetries induced by an A-interchangeable subdomain δ.

Example 3.19 (*Example 3.1 continued*)
Given $MX(T_{gc}, I_{gcin})$, $\{r,g,b\}$ and $\{t,u,v,w\}$ are A-interchangeable subdomains for $A = \{C|1, Color|0\}$. For $B = \{V|1, Edge|1, Edge|2, Color|1, x_1|0, y_1|0, x_2|0, y_2|0, x_3|0\}$, $\{r,g,b\}$ is a B-interchangeable subdomain. However, as $\sigma^B_{(t\ u)}$ does not preserve the interpretation of $Edge$, $\{t,u,v,w\}$ is not a B-interchangeable subdomain. ▽

Many problems, when modelled as a model expansion problem, exhibit subdomain interchangeability. For instance, a set of nurses in a scheduling problem, a set of colors in a graph coloring problem, or a set of trucks in a planning problem often are interchangeable subdomains.

Subdomain interchangeability groups contain a number of symmetries factorial in the size of the interchangeable subdomain, leading to an exponential slowdown of many combinatorial search algorithms. However, as we show in Section 4, many subdomain interchangeability groups can be completely broken with a number of constraints linear in the size of the subdomain.

3.4 More symmetry

Even though local domain symmetry is a useful form of symmetry, it does not capture all symmetry properties that might be present in a model expansion problem.

Example 3.20 (*Example 3.1 continued*)
The graph coloring problem $MX(T_{gc}, I_{gcin})$ asks to color a circular directed graph of 4 vertices $\{t,u,v,w\}$. Note that given any satisfying coloring for this graph, swapping the colors of t and v (or u and w) keeps T_{gc} satisfied. This is a clear symmetry property of the graph coloring instance, but it cannot be captured using the notion of local domain symmetry as defined in this paper. For instance, if we take the argument position set $A = \{V|1, Edge|1, Edge|2, Color|1\}$ representing symmetry in the vertices, then the induced structure transformation $\sigma^A_{(t\ v)}$ is not a symmetry of $MX(T_{gc}, I_{gcin})$ since it does not preserve the interpretation of $Edge$.

One way to fix this is based on the observation that argument positions $Edge|1$ and $Edge|2$ are indistinguishable in T_{gc}: in each sentence of T_{gc}, one could swap any quantifier over $Edge|1$ with one over $Edge|2$, ending up with a sentence equivalent to the original one. In more bold words, argument positions $Edge|1$ and $Edge|2$ *themselves* are symmetric. This symmetry property can be captured by generalizing the notion of an A,π-induced structure transformation to allow for swaps or permutations of argument positions.

For instance, we could define $\sigma_A^{(Edge|1\ Edge|2)(t\ v)}$ to first map $Edge^{I_{gcin}}$ to $\{(e,d) \mid (d,e) \in Edge^{I_{gcin}}\}$ before applying $\sigma_A^{(t\ v)}$. Note that $\sigma_A^{(Edge|1\ Edge|2)(t\ v)}$ would preserve $Edge^{I_{gcin}}$ and I_{gcin} in general, while also preserving satisfaction to T_{gc}, making it a symmetry of $MX(T_{gc}, I_{gc})$. ▽

Similarly, problems with spatial properties often have rotational or reflectional symmetry, which is not covered by the presented notion of local domain symmetry. One such example is the N-Queens problem, which is experimentally investigated in Section 6.

4 Symmetry breaking and local domain interchangeability

A standard approach of dealing with symmetry extends the theory with *symmetry breaking constraints* that eliminate symmetric solutions while guaranteeing that at least one solution to the original problem is preserved (if it exists). This way, a search algorithm will not get stuck in

parts of the search space symmetric to those already explored. A set of symmetry breaking constraints φ is *sound* for a symmetry group $\mathbb{G}$ if for each solution I, there exists *at least* one $\sigma \in \mathbb{G}$ such that $\sigma(I)$ satisfies φ; it is *complete* if there exists *at most* one such $\sigma \in \mathbb{G}$ (Walsh 2012).

Often, symmetry breaking is done by defining a lexicographical order over the set of candidate solutions. For a given symmetry, so-called *lex-leader constraints* then encode that each solution's symmetrical image cannot be strictly smaller under the defined lexicographical order. As long as the chosen lexicographical order is fixed, the conjunction of lex-leader constraints for any set of symmetries is sound.

In a model expansion context, the set of candidate solutions $\Gamma_D^{\Sigma_{out}}$ consists of all Σ_{out}-structures with domain D. A logical formula that is added to the theory takes over the role of a constraint. We construct a lexicographical order $\preceq_\Gamma$ over $\Gamma_D^{\Sigma_{out}}$ from an order $\preceq_D$ over D and an order $\preceq_{\Sigma_{out}}$ over Σ_{out}. Then, $I_{out} \prec_\Gamma I_{out}{}'$ iff there exists some symbol $S \in \Sigma_{out}$ and domain element tuple $\bar{d}$ such that $\bar{d} \notin S^{I_{out}}$, $\bar{d} \in S^{I_{out}{}'}$, and for all $\bar{d}' \prec_D \bar{d}$ it holds that $\bar{d}' \in S^{I_{out}} \Leftrightarrow \bar{d}' \in S^{I_{out}{}'}$, and for all $S' \prec_{\Sigma_{out}} S$, it holds that $S'^{I_{out}} = S'^{I_{out}{}'}$. For the remainder of this section, we leave the order over Σ_{out} implicit, but explicitly state the order $\preceq_D$ over D, as this turns out to be important. Given a model expansion problem with symmetry σ and a lexicographical order over $\Gamma_D^{\Sigma_{out}}$ with $\preceq_D$ as D-order, we use $lex^{\preceq_D}(\sigma)$ to refer to the logical formula encoding the lex-leader constraint for σ. Efficient encodings of the lex-leader constraint into formulas are well-known (Sakallah 2009).

Example 4.1 (*Example 3.1 continued*)
Let $t \prec_D u \prec_D v \prec_D w \prec_D r \prec_D g \prec_D b$ and $A = \{C|1, Color|0\}$. For $MX(T_{gc}, I_{gcin})$, the local domain symmetry $\sigma^A_{(r\ g)}$ is broken by the lex-leader constraint $lex^{\preceq_D}(\sigma^A_{(r\ g)})$, which informally implies that for each vertex v, if all vertices $v' \prec_D v$ are not colored by r or by g, then v cannot be colored with r. Amongst others, this constraint cuts away Σ_{out}-structures that color t with r. ▽

Note that lex-leader constraints are constructed for individual symmetries. In general, to obtain a complete symmetry breaking constraint for a symmetry group $\mathbb{G}$, one needs to post $lex^{\preceq_D}(\sigma)$ for each $\sigma \in \mathbb{G}$. As symmetry groups can contain a factorial amount of symmetries this is infeasible, e.g., in the case of subdomain interchangeability. Instead, the standard approach is *partial symmetry breaking*, where $lex^{\preceq_D}(\sigma)$ is posted for a minimal set of generators σ of $\mathbb{G}$ (Aloul et al. 2006). Partial symmetry breaking is feasible, but does not guarantee that $\mathbb{G}$ is broken completely, leaving symmetrical parts of the search space open to a search engine.

For instance, for a subdomain interchangeability group $\mathbb{G}^A_\delta$, a minimal set of generator symmetries is $\{\sigma^A_{(d\ s(d))} \mid d, s(d) \in \delta\}$, where $s(d)$ is the successor of d in δ according to $\preceq_D$. Other minimal generator sets exist as well, e.g., $\{\sigma^A_{(d_0\ d)} \mid d \in \delta, d \neq d_0\}$ for a fixed $d_0 \in \delta$. However, the choice of the generator set influences the power of the symmetry breaking formula. For subdomain interchangeability groups $\mathbb{G}$, choosing the right generator set can guarantee that the lex-leader constraints used in partial symmetry breaking are actually complete for $\mathbb{G}$:

Theorem 4.2
Let $MX(T, I_{in})$ be a model expansion problem, δ an A-interchangeable subdomain, $\preceq_D$ a total order on domain D and $s(d)$ the successor of d in δ according to $\preceq_D$. If A contains at most one argument position $S|i$ for each symbol $S \in \Sigma_{out}$, then the conjunction of lex-leader constraints

$$\{lex^{\preceq_D}(\sigma^A_{(d\ s(d))}) \mid d, s(d) \in \delta\}$$

is a complete symmetry breaking constraint for the subdomain interchangeability group $\mathbb{G}^A_\delta$.

A strongly related result is that when constructing a relation $R \subseteq D_1 \times \ldots \times D_n$ for which exactly one dimension D_i contains interchangeable values, an efficient lex-leader constraint exists that completely breaks the resulting symmetry (Shlyakhter 2007). Theorem 4.2 can be seen as a conversion of this result to a model expansion context with local domain interchangeability.

Intuitively, Theorem 4.2 states that local domain interchangeability is completely broken by a linear number of lex-leader constraints if the set of argument positions contains at most one argument position for each output symbol. These lex-leader constraints $lex^{\preceq_D}(\sigma^A_{(d\ s(d))})$ are based on swaps $(d\ s(d))$ of two consecutive domain elements over the chosen domain ordering. Note that lex-leader constraints based on swaps of non-consecutive domain elements, e.g., $\{\sigma^A_{(d_0\ d)} \mid d \in \delta, d \neq d_0\}$ for a fixed d_0, do not have this property (Devriendt et al. 2014).

Example 4.3 (*Example 3.1 continued*)
Given the graph coloring problem $MX(T_{gc}, I_{gcin})$, let $A = \{C|1, Color|0\}$ and $r \prec_D g \prec_D b$. $\mathbb{G}^A_{\{r,g,b\}}$ is a subdomain interchangeability group of $MX(T_{gc}, I_{gcin})$. Since A contains only one argument position for the symbol $Color$, it is completely broken by

$$lex^{\preceq_D}(\sigma^A_{(r\ g)}) \wedge lex^{\preceq_D}(\sigma^A_{(g\ b)}) \qquad \triangledown$$

The finite model generation system SEM also breaks this type of symmetry completely, by way of *dynamically* avoiding symmetrical decisions during search. (Zhang and Zhang 1995) The more recent model generator KODKOD (Torlak and Jackson 2007) breaks symmetry statically by posting lex-leader constraints from Aloul et al. (2006) for global domain symmetry. Although Torlak and Jackson (2007) do not mention any completeness result, experiments with a pigeonhole encoding in KODKOD indicate that it uses the right set of generator symmetries to completely break all pigeon and hole interchangeability symmetry.

5 Symmetry detection

In this section, we give two local domain symmetry detection algorithms for model expansion problems. The first detects generators of a local domain symmetry group, the second derives interchangeable subdomains. Both approaches work on a first-order level, avoiding the need to *ground* the model expansion problem to a propositional counterpart. Both algorithms are based on Theorem 3.16, which conditions argument position set A to be connectively closed under decomposition theory I^*_{in}. To find such A, one simply constructs a partition of T^*'s argument positions. Using a disjoint-set data structure[1], the computational cost to find A is linear in the size of T. In the following subsections, we assume a set of argument positions A satisfying the connectedness condition is available, leaving only the concern of finding an appropriate domain permutation π (Section 5.1) or interchangeable subdomain δ (Section 5.2).

5.1 Local domain symmetry detection

Our approach follows other symmetry detection techniques (Aloul et al. 2002; Drescher et al. 2011) by converting the symmetry detection problem to a *graph automorphism* detection problem. An *automorphism* of a graph is a permutation τ of its vertices such that each vertex pair

[1] en.wikipedia.org/wiki/Disjoint-set_data_structure

(v,u) forms an edge iff $(\tau(v),\tau(u))$ forms an edge. If the graph is colored, then each vertex v must have the same color as $\tau(v)$.

This existing work encodes a propositional theory into a graph, which we call the *detection graph*. If the detection graph is well-constructed, its automorphism group corresponds to a symmetry group of the propositional theory. Tools such as SAUCY (Katebi et al. 2010) then are employed to derive generators for the detection graph's automorphism group, which in turn are converted to symmetry generators for the propositional theory.

Our approach differs by not encoding a propositional theory into the detection graph, but an input structure and a set of argument positions, as these are all we need to detect local domain symmetry. Formally, given a structure I and an argument position set A, we construct an undirected colored graph whose automorphisms correspond to domain permutations π such that $\sigma^A_\pi(I) = I$ – satisfying the second condition of Theorem 3.16.

Definition 5.1 (*Domain permutation graph*)
Let I be a Σ-structure with domain D and A a set of argument positions. The *domain permutation graph* $DPG(I,A)$ for I and A is an undirected colored graph with labeled vertices V, edges E and color function C that satisfies the following requirements:

V is partitioned into three subsets: DE (domain element vertices), AP (argument position vertices) and IT (interpretation tuple vertices). DE contains a vertex labeled d for each $d \in D$. AP contains $k+1$ vertices labeled $\{d.i \mid i \in [0..k]\}$ for each $d \in D$, with k the maximum arity of symbols in Σ. IT contains a vertex labeled $S(\bar{d})$ for each tuple $\bar{d} \in S^I$ with $S^I \in I$.

E consists only of edges between DE and AP, and between AP and IT. An AP vertex labeled $d.i$ is connected to a DE vertex e iff $d = e$. An IT vertex labeled $S(\ldots,d_i,\ldots)$ is connected to an AP vertex $e.j$ iff $d = e$, $i = j$ and $S|i \in A$.

Vertices from different partitions have different colors. All DE vertices have the same color. Two AP vertices labeled $d.i$ and $e.j$ have the same color iff $i = j$. Two IT vertices labeled $S(d_1,\ldots,d_n)$ and $R(e_1,\ldots,e_n)$ have the same color iff $S = R$ and $d_i = e_i$ for all i such that $S|i \notin A$.

The intuition behind the domain permutation graph $DPG(I,A)$ is that a permutation of its DE vertices corresponds to a domain permutation π, a permutation of its IT vertices corresponds to a permutation of domain element tuples in interpretations in I, and the AP vertices and vertex coloring serve to link DE and IT in such a way that Definition 3.6 is preserved for automorphisms.

Theorem 5.2
Let I be a Σ-structure with domain D and A a set of argument positions. There exists a bijection between the automorphism group of the domain permutation graph $DPG(I,A)$ and the group of domain permutations π such that $\sigma^A_\pi(I) = I$. This bijection maps an automorphism τ to domain permutation π iff $\tau(d) = \pi(d)$ for all DE vertices (equated with domain elements) d.

Example 5.3 (*Example 3.17 continued*)
Using argument position set $A = \{Edge_1|1, Edge_1|2, Color|1, x_1|0, y_1|0, x_3|0\}$ (which is connectively closed under T^*_{gc}) and input structure I^*_{gcin}, the domain permutation graph $DPG(I^*_{gcin},A)$ is illustrated in Figure 5.1. The automorphism group of $DPG(I^*_{gcin},A)$ corresponds to the group of induced structure transformations σ^A_π such that $\sigma^A_\pi(I^*_{gcin}) = I^*_{gcin}$. As a result, its automorphism group corresponds to a local domain symmetry group of $MX(T_{gc},I_{gcin})$. E.g., $\sigma^A_{(t\ u\ v\ w)}$ corresponds to an automorphism that permutes the four left-most groups of five vertices, and $\sigma^A_{(b\ g)}$ to an automorphism that swaps the two right-most groups of four vertices. ▽

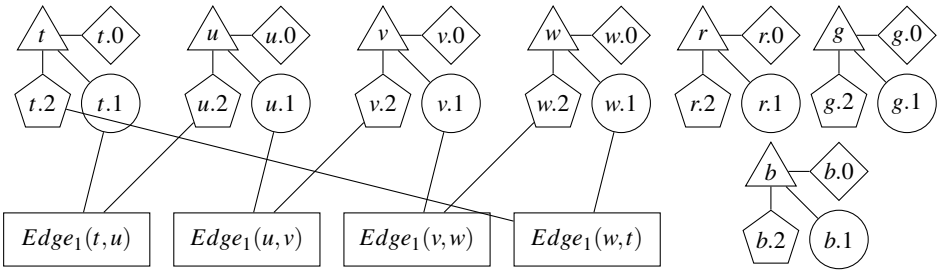

Fig. 1. Domain permutation graph $DPG(I^*_{gcin}, A)$ with $A = \{Edge_1|1, Edge_1|2, Color|1, x_1|0, y_1|0, x_3|0\}$. Each shape denotes a unique color, so vertices with the same shape have the same color.

Let k be the largest arity of a symbol in I for a domain permutation graph $DPG(I,A)$. The size of DE is $|D|$, the size of AP is $(k+1)|D|$, and the size of IT is $|I|$, which is $O(|D|^k)$. Thus, the total number of nodes is $O(k|D| + |D|^k)$. There are $(k+1)|D|$ edges between DE and AP, and, if all argument positions over some symbol S/k occur in A, then there are $O(k|I|) = O(k|D|^k)$ edges between AP and IT. Thus, the total number of edges is $O(k|D|^k)$.

Note that the size of $DPG(I,A)$ does not depend on the size of the theory of the model expansion problem. This is a major advantage compared to automorphism-based symmetry detection on ground theories, as the detection graph grows linearly with the ground theory (Drescher et al. 2011), which is typically much larger than the input structure.

5.2 Subdomain interchangeability detection

While the previous subsection detects local domain symmetry generators individually, it is not clear what type of symmetry group they form. To optimally construct symmetry breaking constraints for symmetry groups, we need to detect subdomain interchangeability as well. Then, by Theorem 4.2, we will often be able to break subdomain interchangeability groups completely with a set of lex-leader constraints linear in $|D|$.

Given a model expansion problem $MX(T, I_{in})$ with decomposition $MX(T^*, I^*_{in})$ and a set of argument positions A connectively closed under T^*, the task at hand is to find a subdomain $\delta \subseteq D$ such that for each permutation π over δ, $\sigma^A_\pi(I^*_{in}) = I^*_{in}$. If so, Theorem 3.16 guarantees σ^A_π to be a symmetry of $MX(T, I_{in})$, which makes δ an A-interchangeable subdomain.

The actual algorithm finds a partition Δ of D, such that each $\delta \in \Delta$ is A-interchangeable. The idea is based on the fact that the permutation group of a set is generated by swaps of two elements of the set. As such, if we know for each pair $d_1, d_2 \in D$ whether $\sigma^A_{(d_1\ d_2)}(I^*_{in}) = I^*_{in}$, it is straightforward to construct the partition Δ. The resulting symmetry detection algorithm is simple: for each pair of domain elements $d_1, d_2 \in D$, check whether $\sigma^A_{(d_1\ d_2)}(I^*_{in}) = I^*_{in}$. When using a disjoint-set data structure[1] to keep track of the partition Δ, the complexity of this algorithm is $O(|D|^2 |I^*_{in}|)$. The algorithm can be optimized by exploiting transitivity, domain element occurrence counting or unary symbols partitioning the domain, but this does not improve the worst-case complexity.

Example 5.4 (*Example 3.17 continued*)
Given argument position set $A = \{C_1|1, Color|0\}$ (which is connectively closed under T^*_{gc}), we detect A-interchangeable domains by checking whether the (only) input symbol C_1 has the same interpretation in $\sigma^A_{(d_1\ d_2)}(I^*_{in})$ as in I^*_{in} for combinations of $d_1, d_2 \in \{t, u, v, w, r, g, b\}$. For $(d_1, d_2) \in$

$\{(t,u),(u,v),(v,w),(r,g),(g,b)\}$ this is indeed the case. For $(d_1,d_2)=(w,r)$ this is not the case, so the A-interchangeable sets are $\{t,u,v,w\}$ and $\{r,g,b\}$. ▽

6 Experiments

Based on the theory and algorithms presented in this paper, we implemented symmetry exploitation in the model expansion inference of the IDP system (De Cat et al. 2016). IDP is a *knowledge base system* where knowledge about a problem can be modelled in FO($\cdot$), a rich extension of first-order logic (Denecker and Ternovska 2008). Our implementation (including source code) is available online[2], and is incorporated in IDP version 3.6.0 and up. Our implementation makes use of SAUCY version 3 to solve the graph automorphism component of symmetry detection, and constructs symmetry breaking formulas based on the lex-leader encoding of Aloul et al. (2006).

We compare this implementation with the ASP system CLASP (Gebser et al. 2014) version 3.1.4, using version 4.5.4 of the ASP grounder GRINGO to generate ground answer set programs. For CLASP, the symmetry breaking preprocessor SBASS has been developed (Drescher et al. 2011). SBASS takes a ground answer set program, encodes it to a detection graph, uses SAUCY to solve the automorphism detection problem, converts SAUCY's output to permutations of propositional atoms that induce symmetries, and constructs symmetry breaking constraints following Aloul et al. (2006).

Our experiment uses four different system configurations: IDP and CLASP refer to both systems without symmetry breaking, IDPSYM refers to IDP extended with the techniques described in this paper, and SBASS refers to CLASP coupled with the symmetry breaking preprocessor.

This experiment can only broadly compare the IDP and CLASP configurations, as both systems use similar but ultimately different techniques to solve the model expansion problem. Our main interest is to investigate the types of symmetry detected, the overhead needed to detect those, and the relative speedup gained when activating symmetry algorithms for both systems. We expect that IDPSYM, compared to SBASS, has less symmetry detection overhead, as IDPSYM detects symmetry on the first-order level instead of on the ground level. E.g., the structure information present in a set of connectively closed argument positions can be derived with a syntactical check on the first-order theory, but this information is lost after grounding. As a result, we expect IDPSYM's detection graph to be smaller, or even non-existent.[3] Also, we expect a larger relative speedup for IDPSYM than for SBASS on problems with a lot of subdomain interchangeability, as only IDPSYM detects and completely breaks this type of symmetry. However, as mentioned in Section 3.4, IDPSYM's detected symmetry group might be smaller than SBASS's, as not all symmetry properties of a problem can be captured by our notion of local domain symmetry.

Our benchmark set consists of four problem families: **pigeons**, **crew**, **graceful** and **200queens**. **pigeons** is a set of 16 unsatisfiable pigeonhole instances where n pigeons must be placed in $n-1$ different holes. The pigeons and holes are indistinguishable, leading to subdomain interchangeability symmetry groups. **crew** is a set of 42 unsatisfiable airline crew scheduling instances, where optimality has to be proven for a minimal crew assignment given a moderately complex flight plan. The instances are generated by hand, with the number of crew members ranging

[2] `bitbucket.org/krr/fo-sym-experiments`

[3] IDPSYM does not construct the detection graph if the only generators it will detect are due to subdomain interchangeability. Given an argument position set A, this is the case if for each symbol S, A contains at most one argument position over S.

from 5 to 25. Crew members have different attributes, but depending in the instance, multiple crew members exist with exactly the same attribute set, leading to subdomain interchangeability symmetry. **graceful** consists of 60 graceful graph instances, taken from 2013's ASP competition. These instances require to label a graph's vertices and edges such that all vertices have a different label, all edges have a different label, and each edge's label is the difference of the labels of the vertices it connects. The labels used are $\{0,1,\ldots,n\}$, with n the number of edges. Any symmetry exhibited by the input graph is present, as well as a symmetry mapping each vertex' label l to $n-l$. **200queens** is one N-Queens instance trying to fit 200 queens on a 200 by 200 chessboard so that no queen threatens another. The symmetry present in **200queens** is due to the rotational and reflective symmetries of the chessboard.

The available resources were 6GB RAM and 1000s timeout on an Intel® Xeon® E3-1225 CPU with Ubuntu 14.04 Linux kernel 3.13 as operating system. FO(·) and ASP specifications, instances and detailed experimental results are available online.[2] Table 1 summarizes the results.

	CLASP	SBASS				IDP	IDPSYM				
	#	#	t	V	π	#	#	t	V	π	δ
pigeons (16)	8	11	50.9	48814	43.5	8	16	0.0	0 [3]	0	2
crew (42)	32	36	0.0	1722	7.8	28	39	0.0	0 [3]	0	4.1
graceful (60)	33	20	0.7	127860	5.5	26	13	0.4	15201	5.4	0.6
200queens (1)	1	1	76.2	9357802	2	1	1	6.8	0 [3]	0	0

Table 1. *Experimental results of* CLASP- *and* IDP-*based solvers with and without symmetry breaking. # represents the number of solved instances, t the average symmetry detection time in seconds, V the average number of vertices in the detection graph, π the average number of symmetry generators detected by* SAUCY*, and δ the average number of interchangeable subdomains detected.*

When analyzing the results on **pigeons**, it is clear that plain CLASP and IDP get lost in symmetric parts of the search tree, solving only 8 instances (up to 12 pigeons). SBASS can only solve three more instances (up to 15 pigeons), as the derived symmetry generators do not suffice to construct strong symmetry breaking constraints. These results are consistent with Drescher et al. (2011). IDPSYM detects the pigeon and hole interchangeable subdomains, and its complete symmetry breaking constraints allow all 16 instances to be solved (up to 100 pigeons). As far as symmetry detection time goes, unlike SBASS, IDPSYM has negligible detection overhead.

The results on **crew** are similar to **pigeons** but less outspoken. The reason is that even though there are more subdomain interchangeability groups, the subdomains are a lot smaller, incurring less symmetry overhead. As a result, IDPSYM only enjoys a small advantage over SBASS, but does reverse the situation where pure CLASP outperforms pure IDP. Concerning symmetry detection, IDPSYM has to analyze the input structure before deriving any subdomain interchangeability groups, which contrasts with the trivially interchangeable pigeons and holes in **pigeons**. Nonetheless, IDPSYM solves this task in the blink of an eye, as does SBASS.

Continuing with **graceful**, it is striking that the number of solved instances is *reduced* by symmetry breaking. Upon closer inspection, this is only the case for satisfiable instances. For unsatisfiable **graceful** instances, SBASS solves two more than CLASP, and IDPSYM solves three more than IDP. This discrepancy is not uncommon, as static symmetry breaking reduces the search space by removing possibly easy-to-find solutions. These results are also consistent with those reported by Drescher et al. (2011). Looking at the number of symmetry generators detected,

both SBASS and IDPSYM detect about the same number of symmetry generators, indicating that they detect the same symmetry group. Note that IDPSYM also detects a few subdomain interchangeability groups, apparently present in the input graph. As far as symmetry detection overhead goes, SBASS is slower than IDPSYM. This is not surprising, as SBASS' detection graph is almost an order of magnitude larger than IDPSYM's. We conclude that for **graceful**, IDPSYM detects the same symmetry group as SBASS, though with less overhead.

Lastly, for **200queens**, IDPSYM cannot detect the geometric symmetries of the chessboard, as this type of symmetry does not fit the definition of local domain symmetry. For instance, a square (i, j) on the chess board is diagonally reflected to square (j, i), while square (i, k) is reflected to (k, i). Domain element i is mapped to both j and k at position 0, violating the local domain symmetry requirement that it stems from a domain *permutation*. SBASS can detect this type of symmetry, as it detects permutations of ground atoms instead of domain elements. However, note the significant overhead incurred, as the detection graph is huge.

We conclude that our approach has very low symmetry detection overhead, due to a smaller or non-existent detection graph. Moreover, by completely breaking subdomain interchangeability, we significantly increase the number of solved instances. However, not all symmetry present in the problem set is detected by our approach.

7 Conclusion

We presented the notion of local domain symmetry for model expansion problems, which manifests itself on the first-order level. We gave a completeness result on the strength of symmetry breaking constraints for a special case of local domain symmetry, and we posted syntactical conditions to efficiently detect symmetry from a model expansion specification. Our experiment highlights the strengths and weaknesses of our approach. We have a very low symmetry detection overhead and we give symmetry breaking completeness guarantees for local domain interchangeability that are effective in practice. However, we cannot detect some forms of symmetry.

It is worth mentioning that local domain symmetry is not limited to pure classical logic; it is straightforward to extend our work to cardinalities, types or arithmetic. Similarly, logic programs under stable or well-founded semantics have symmetry properties induced by permutations of domain elements (or Herbrand constants) and sets of argument positions. Our work easily transfers to these domains. In fact, our implementation in IDP already supports such extensions.

Investigating which types of symmetry fall outside our formalism, and inventing ways to detect and exploit these types of symmetry is interesting future work. One idea is that not only permutations of the domain lead to symmetry, but permutations on (argument positions of) symbols in the vocabulary do as well. We suspect there is also room for improvement on the symmetry breaking front. Limiting the size of individual symmetry breaking constraints is known to improve their performance. Also, to avoid deteriorating performance on satisfiable instances, one should look into dynamic symmetry breaking approaches, as these typically do not cut away solutions a priori.

8 Acknowledgements

This research was supported by the project GOA 13/010 Research Fund KU Leuven and projects G.0489.10, G.0357.12 and G.0922.13 of FWO (Research Foundation - Flanders). Bart Bogaerts

is supported by the Finnish Center of Excellence in Computational Inference Research (COIN) funded by the Academy of Finland (grant #251170).

References

ALOUL, F., RAMANI, A., MARKOV, I., AND SAKALLAH, K. 2002. Solving difficult SAT instances in the presence of symmetry. In *Design Automation Conference, 2002. Proceedings. 39th*. 731–736.

ALOUL, F. A., SAKALLAH, K. A., AND MARKOV, I. L. 2006. Efficient symmetry breaking for Boolean satisfiability. *IEEE Transactions on Computers 55,* 5, 549–558.

AUDEMARD, G. AND BENHAMOU, B. 2002. Reasoning by symmetry and function ordering in finite model generation. In *Automated Deduction - CADE-18, 18th International Conference on Automated Deduction, Copenhagen, Denmark, July 27-30, 2002, Proceedings*, A. Voronkov, Ed. Lecture Notes in Computer Science, vol. 2392. Springer, 226–240.

CLAESSEN, K. AND SÖRENSSON, N. 2003. New Techniques that Improve MACE-style Model Finding. In *Workshop on Model Computation (MODEL)*.

DE CAT, B., BOGAERTS, B., BRUYNOOGHE, M., JANSSENS, G., AND DENECKER, M. 2016. Predicate logic as a modelling language: The IDP system. *CoRR abs/1401.6312v2*.

DENECKER, M. AND TERNOVSKA, E. 2008. A logic of nonmonotone inductive definitions. *ACM Trans. Comput. Log. 9,* 2 (Apr.), 14:1–14:52.

DEVRIENDT, J., BOGAERTS, B., AND BRUYNOOGHE, M. 2014. BreakIDGlucose: On the importance of row symmetry in SAT. In *Proceedings of the Fourth International Workshop on the Cross-Fertilization Between CSP and SAT (CSPSAT)*.

DRESCHER, C., TIFREA, O., AND WALSH, T. 2011. Symmetry-breaking answer set solving. *IA Communications 24,* 2, 177–194.

ENDERTON, H. B. 2001. *A Mathematical Introduction To Logic*, Second ed. Academic Press.

FLENER, P., FRISCH, A. M., HNICH, B., KIZILTAN, Z., MIGUEL, I., PEARSON, J., AND WALSH, T. 2002. Breaking row and column symmetries in matrix models. In *Principles and Practice of Constraint Programming - CP 2002*, P. Hentenryck, Ed. LNCS, vol. 2470. Springer Berlin Heidelberg, 462–477.

GEBSER, M., KAMINSKI, R., KAUFMANN, B., AND SCHAUB, T. 2014. *Clingo* = ASP + control: Preliminary report. In *Technical Communications of the Thirtieth International Conference on Logic Programming (ICLP'14)*, M. Leuschel and T. Schrijvers, Eds. Vol. 14(4-5). Online Supplement.

GENT, I. P., PETRIE, K. E., AND PUGET, J.-F. 2006. Symmetry in constraint programming. *Handbook of Constraint Programming 10*, 329–376.

KATEBI, H., SAKALLAH, K. A., AND MARKOV, I. L. 2010. Symmetry and satisfiability: An update. In *SAT*, O. Strichman and S. Szeider, Eds. LNCS, vol. 6175. Springer, 113–127.

SAKALLAH, K. A. 2009. *Symmetry and Satisfiability*. Frontiers in Artificial Intelligence and Applications, vol. 185. IOS Press, Chapter 10, 289–338.

SHLYAKHTER, I. 2007. Generating effective symmetry-breaking predicates for search problems. *Discrete Appl. Math. 155,* 12 (June), 1539–1548.

TORLAK, E. AND JACKSON, D. 2007. Kodkod: A relational model finder. In *TACAS*, O. Grumberg and M. Huth, Eds. LNCS, vol. 4424. Springer, 632–647.

WALSH, T. 2012. Symmetry breaking constraints: Recent results. *CoRR abs/1204.3348*.

ZHANG, J. AND ZHANG, H. 1995. Sem: A system for enumerating models. In *Department of Philosophy University of Wisconsin-Madison Mathematics and Computer Science*. 298–303.

Appendix A Proofs

Theorem 3.10
Let Σ be a vocabulary, T a theory over Σ, π a domain permutation and A a set of argument positions. If A is connectively closed under T, then σ_π^A is a local domain symmetry of T.

Proof
To prove this theorem, we prove the following consecutive claims for each Σ-structure I. Without loss of generalization, we assume I interprets the neccessary variables.

1. For each term $f(\bar{t})$ that occurs in T, it holds that $f(\bar{t})^{\sigma_\pi^A(I)} = \begin{cases} \pi(f(\bar{t})^I) & \text{if } f|0 \in A \\ f(\bar{t})^I & \text{otherwise} \end{cases}$
2. For each atom a of the form $P(t_1,\ldots,t_n)$ or of the form $t_1 = t_2$ that occurs in T, it holds that $a^{\sigma_\pi^A(I)} = a^I$.
3. For each formula φ that occurs in T, $\varphi^{\sigma_\pi^A(I)} = \varphi^I$.

The first claim is proven by induction on the subterm relation. The induction step follows from the fact that A is connectively closed. The second claim follows from the first, also using the fact that A is connectively closed. Consider for instance the case of atom $f(\bar{t}) = g(\bar{t}')$ occurring in T, with $f|0 \in A$. Then $g|0 \in A$ (since A is connectively closed), so $f(\bar{t})^{\sigma_\pi^A(I)} = g(\bar{t}')^{\sigma_\pi^A(I)}$ iff $\pi(f(\bar{t})^I) = \pi(g(\bar{t}')^I)$ iff $f(\bar{t})^I = g(\bar{t}')^I$ (since π is a permutation). The other cases are analogous.

The last claim follows by induction on the subformula relation since the value of a first-order formula is entirely determined by the value of the atoms occuring in it. Consider for instance the case of formula $\exists x\colon \varphi$ occurring in T with $x|0 \in A$. $(\exists x\colon \varphi)^I$ holds iff there exists a $d \in D$ such that $\varphi^{I[x:d]}$ holds. By the induction hypothesis, $\varphi^{I[x:d]} = \varphi^{\sigma_\pi^A(I[x:d])} = \varphi^{\sigma_\pi^A(I)[x:\pi(d)]}$ (since $x|0 \in A$). $(\exists x\colon \varphi)^{\sigma_\pi^A(I)}$ holds iff there exists a $d' \in D$ such that $\varphi^{\sigma_\pi^A(I)[x:d']}$ holds. Without loss of generalization, let $d' = \pi(d)$, then $(\exists x\colon \varphi)^I = (\exists x\colon \varphi)^{\sigma_\pi^A(I)}$. The other cases are analogous. □

Theorem 3.16
Let $MX(T, I_{in})$ be a model expansion problem with decomposition $MX(T^*, I_{in}^*)$. If A is connectively closed under T^* and $\sigma_\pi^A(I_{in}^*) = I_{in}^*$ then σ_π^A is a symmetry for $MX(T, I_{in})$.

Proof
Since $MX(T, I_{in})$ and $MX(T^*, I_{in}^*)$ have the same set of solutions, it suffices to prove that σ_π^A is a symmetry of $MX(T^*, I_{in}^*)$. Firstly, due to the connectively closed condition, σ_π^A is a symmetry of T^*, so $I_{in}^* \sqcup I_{out} \models T^*$ iff $\sigma_\pi^A(I_{in}^* \sqcup I_{out}) \models T^*$. Secondly, since $\sigma_\pi^A(I_{in}^*) = I_{in}^*$, $I_{in}^* \sqcup I_{out} \models T^*$ iff $I_{in}^* \sqcup \sigma_\pi^A(I_{out}) \models T^*$, so σ_π^A is a symmetry for $MX(T^*, I_{in}^*)$. □

Theorem 4.2
Let $MX(T, I_{in})$ be a model expansion problem, δ an A-interchangeable subdomain, $\preceq_D$ a total order on domain D and $s(d)$ the successor of d in δ according to $\preceq_D$. If A contains at most one argument position $S|i$ for each symbol $S \in \Sigma_{out}$, then the conjunction of lex-leader constraints

$$\{lex^{\preceq_D}(\sigma^A_{(d\ s(d))}) \mid d \in \delta\}$$

is a complete symmetry breaking constraint for the subdomain interchangeability group $\mathbb{G}_\delta^A$.

Proof
To prove this theorem, we (1) convert the task of finding a solution to a model expansion problem to a constraint programming problem, where an assignment over a set of Boolean variables

V has to be found. Further, we show that (2) a subset of these Boolean variables can be organized as a matrix M_δ, where each permutation over the rows of M_δ corresponds to a permutation over δ. The interchangeability group $\mathbb{G}_\delta^A$ then corresponds to a row interchangeability symmetry group induced by permuting M_δ's rows. Using a result from constraint programming, such row interchangeability symmetry groups are broken completely by posting a lex-leader constraint (based on the appropriate row ordering) for each symmetry induced by the swap of two consecutive rows (Flener et al. 2002; Devriendt et al. 2014). This corresponds to posting $\{lex^{\preceq_D}(\sigma^A_{(d\ s(d))}) \mid d \in \delta\}$, ending the proof.

(1) Given a vocabulary Σ_{out} and a domain D, finding a Σ_{out}-structure consists of deciding for each $\bar{d} \in D^n$ whether $\bar{d} \in S^{I_{out}}$ for each symbol $S/n \in \Sigma_{out}$. Hence, a model expansion problem $MX(T, I_{in})$ can be seen as finding an assignment to a set of Boolean variables $V = \{S(\bar{d}) \mid S/n \in \Sigma_{out}, \bar{d} \in D^n\}$ such that $I_{in} \sqcup I_{out} \models T$. A local domain symmetry σ^A_π for $MX(T, I_{in})$ now corresponds to a *variable symmetry* (Flener et al. 2002) mapping

$$S(d_1, \ldots, d_n) \quad \text{to} \quad S(\tau_{S|1}(d_1), \ldots, \tau_{S|n}(d_n))$$

where $\tau_{S|i}(d) = \pi(d)$ if $S|i \in A$ and $\tau_{S|i}(d) = d$ otherwise.

(2) The variables in V that are not fixed by some $\sigma^A_\pi \in \mathbb{G}^A_\delta$ are those $S(\ldots, d_{j-1}, \delta_i, d_{j+1}, \ldots)$ where $\delta_i \in \delta$ is the jth domain element of an S-tuple with $S|j \in A$. We can partition this subset into "rows" $R_{\delta_i} = \{S(\ldots, d_{j-1}, \delta_i, d_{j+1}, \ldots) \mid S|j \in A, d_k \in D\}$ where δ_i is fixed. It is clear that $\sigma^A_\pi(R_{\delta_i}) = R_{\pi(\delta_i)}$, so a permutation of the set of rows corresponds to a symmetry of $\mathbb{G}^A_\delta$. Since A contains at most one argument position for each $S \in \Sigma_{out}$, these rows are pairwise disjoint, and under some column organization form the requested matrix M_δ. □

Theorem 5.2
Let I be a Σ-structure with domain D and A a set of argument positions.

There exists a bijection between the automorphism group of the domain permutation graph $DPG(I, A)$ and the group of domain permutations π such that $\sigma^A_\pi(I) = I$. This bijection maps an automorphism τ to domain permutation π iff $\tau(d) = \pi(d)$ for all DE vertices (equated with domain elements) d.

Proof
We prove the bijection by showing that all induced structure transformations σ^A_π with $\sigma^A_\pi(I_{in}) = I_{in}$ correspond to an automorphism of $DPG(I_{in}, A)$ ($\Rightarrow$) and vice versa ($\Leftarrow$).

First, some preliminaries. For a given symbol S, let each tuple $(d_1, \ldots, d_n) \in S^{I_{in}}$ be split as two tuples $d_A^+ d_A^-$ such that $d_A^+ = \{d_i \mid S|i \in A\}$ and $d_A^- = \{d_i \mid S|i \notin A\}$. Let π naturally extend to tuples: $\pi((d_1, \ldots, d_n)) = (\pi(d_1), \ldots, \pi(d_n))$. The symmetrical interpretation $\sigma^A_\pi(I_{in})$ can then be described as $\{\pi(d_A^+) d_A^- \mid d_A^+ d_A^- \in S^{I_{in}}\}$, so $\sigma^A_\pi(I_{in}) = I_{in}$ iff for all symbols S, $d_A^+ d_A^- \in S^{I_{in}}$ iff $\pi(d_A^+) d_A^- \in S^{I_{in}}$. Also, without loss of generalization, let an IT vertex's label be $S(d_A^+ d_A^-)$ for symbol S. Lastly, $(v, w) \in E$ denotes that graph E has an (undirected) edge between vertices v and w.

($\Rightarrow$) If $\sigma^A_\pi(I_{in}) = I_{in}$, σ^A_π corresponds to a permutation α of the vertices of $DPG(I_{in}, A)$: $\alpha(d) = \pi(d)$ (for DE vertices), $\alpha(d.i) = \pi(d).i$ (for AP vertices), $\alpha(S(d_A^+ d_A^-)) = S(\pi(d_A^+) d_A^-)$ (for IT vertices). We show that α is an automorphism of $DPG(I_{in}, A)$.

By the definition of $DPG(I_{in}, A)$, α preserves the colors. To show that α preserves the edges, we need to show that $(v, w) \in DPG(I_{in}, A)$ iff $(v, w) \in \alpha(DPG(I_{in}, A))$. Firstly, remark that α maps each vertex in a layer to another vertex in that layer, so we only need to check whether the edges between (1) DE-AP and (2) AP-IT are conserved.

(1) The following statements are equivalent

$$\begin{aligned}
&(\alpha(d),\alpha(e.i)) \in \alpha(DPG(I_{in},A)) \\
&\quad (d,e.i) \in DPG(I_{in},A) && (\alpha \text{ is a permutation of vertices}) \\
&\quad d = e && (\text{definition of domain permutation graph}) \\
&\quad \pi(d) = \pi(e) && (\pi \text{ is a permutation}) \\
&\quad (\pi(d),\pi(e).i) \in DPG(I_{in},A) && (\text{definition of domain permutation graph}) \\
&\quad (\alpha(d),\alpha(e.i)) \in DPG(I_{in},A) && (\text{definition of } \alpha)
\end{aligned}$$

(2) Similarly, the following statements are equivalent

$$\begin{aligned}
&(\alpha(d.i),\alpha(S(d_A^+ d_A^-)) \in \alpha(DPG(I_{in},A)) \\
&\quad (d.i,S(d_A^+ d_A^-)) \in DPG(I_{in},A) && (\alpha \text{ is a permutation of vertices}) \\
&\quad d_i \in d_A^+ && (\text{definition of domain permutation graph}) \\
&\quad \pi(d_i) \in \pi(d_A^+) && (\pi \text{ is a permutation}) \\
&\quad (\pi(d).i,S(\pi(d_A^+)d_A^-)) \in DPG(I_{in},A) && (\text{definition of domain permutation graph}) \\
&\quad (\alpha(d.i),\alpha(S(d_A^+ d_A^-)) \in DPG(I_{in},A) && (\text{definition of } \alpha)
\end{aligned}$$

($\Leftarrow$) We must show that an automorphism α of $DPG(I_{in},A)$ corresponds to an A,π-induced structure transformation σ_π^A such that $\sigma_\pi^A(I_{in}) = I_{in}$.

Notice that, since α is an automorphism of a three-layered graph with different colors for each layer DE, AP and IT, we can write it as a composition of three permutations $\alpha_{DE} \circ \alpha_{AP} \circ \alpha_{IT}$. As there exists a bijection between DE and the domain D of I_{in}, we assume $\alpha_{DE} \backsimeq \pi$, with π a permutation of D.

We now show that (1) $\alpha(d.i) = \pi(d).i$ and (2) $\alpha(S(d_A^+ d_A^-)) = S(\pi(d_A^+)d_A^-)$. From this, it follows that α represents a structure transformation σ_π^A mapping tuples $d_A^+ d_A^-$ to $\pi(d_A^+)d_A^-$, and hence, $\sigma_\pi^A(I_{in}) = I_{in}$.

(1) Since $d.i$ and $e.j$ have the same color iff $i = j$, $\alpha(d.i) = e.i$ for some domain element e. As each vertex $d.i$ is connected to exactly one vertex d, $\alpha(d.i) = \pi(d).i$.

(2) Since $S(d_A^+ d_A^-)$ and $R(e_A^+ e_A^-)$ have the same color iff $S = R$ and $d_A^- = e_A^-$, $\alpha(S(d_A^+ d_A^-)) = S(e_A^+ d_A^-)$ for some tuple domain elements e. All that is left to show is that $e_A^+ = \pi(d_A^+)$. For this, note that $S(d_A^+ d_A^-)$ is connected only to $d.i$ for each d on index i in d_A^+. As α is an automorphism that maps $d.i$ to $\pi(d).i$, $\alpha(S(d_A^+ d_A^-)$ must be connected only to all $\pi(d).i$. The only vertex doing so (taking colors into account) is $S(\pi(d_A^+)d_A^-)$. $\square$